Depth-first search (DFS) traverses a directed graph $G$. The defining characteristic of this traversal is that, whenever it visits a vertex $u$, it recursively traverses the graph starting at each $v \in \text{Adj}[u]$ before finishing $u$. To put it another way, DFS replaces the FIFO queue of BFS by a LIFO stack.

DFS assigns each vertex $u$ in $G$ two values: a starting time $s[u]$ and a finishing time $f[u]$. When the search first discovers $u$, it sets $u$’s start time; when it finishes traversing all the edges out of $u$, it sets $u$’s finishing time. Time is a global variable in the DFS algorithm that is incremented whenever a vertex’s starting or finishing time is set. To ensure that every vertex in $G$ is assigned starting and finishing times, DFS loops over each vertex of $G$ (in some arbitrary order), starting the recursive traversal from every vertex that has not yet been visited.

We talk about vertices being in one of several states, which can be identified by looking at their starting and finishing times. Initially, all these times are set to 0; the traversal gradually assigns them values $> 0$.

- If a vertex $u$ has $s[u] = f[u] = 0$, we say that it is undiscovered. In other words, it has not yet been visited by the algorithm.
- If $s[u] > 0$ but $f[u] = 0$, we say that $u$ is in-progress. It has been discovered, but not yet finished.
- If $f[u] > 0$, we say that $u$ is finished.
DFS($G$)

for $u \in V$ do

\hspace{1em} s[u] \leftarrow 0
\hspace{1em} f[u] \leftarrow 0
\hspace{1em} parent[u] \leftarrow \text{null}

time \leftarrow 1

for $u \in V$ do

\hspace{1em} if $s[u] = 0$ \hspace{1em} $\triangleright$ undiscovered

\hspace{2em} DFSVisit($G$, $u$)

DFSVisit($G$, $u$)

\hspace{1em} s[u] \leftarrow \text{time} \hspace{1em} $\triangleright$ start $u$

\hspace{2em} time++

for $v \in \text{Adj}[u]$ do

\hspace{2em} if $s[v] = 0$ \hspace{1em} $\triangleright v$ not visited yet

\hspace{3em} parent[v] \leftarrow u

\hspace{3em} DFSVisit($G$, $v$) \hspace{1em} $\triangleright$ recur before continuing adj list

\hspace{2em} f[u] \leftarrow \text{time} \hspace{1em} $\triangleright$ finish $u$

\hspace{2em} time++
Here are descriptions of the two applications we studied for DFS: detecting whether a directed graph $G$ is cyclic, and computing a topological ordering on $G$’s vertices.

**Cycle Detection**($G$)

1. Run DFS on $G$.
2. If DFS follows an edge $(u, v)$ and discovers that $v$ is gray (i.e. that $s[v] > 0$ but $f[v] = 0$), return “cyclic”.
3. Otherwise, return “acyclic”.

Recall that a topological ordering on a directed, acyclic graph $G$ is a listing of $G$’s vertices, such that if edge $(u, v)$ exists in $G$, $u$ comes before $v$ in the list.

**TopologicalSort**($G$)

1. Run DFS on $G$.
2. If DFS reports “cyclic,” then fail (graphs with directed cycles have no topological ordering).
3. Otherwise, output the vertices of $G$ in *descending order* by their finishing times $f[v]$. 